IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Application Serial No. 10/087,027
Filing Date .... February 28, 2002
Inventorship... Adam W. Smith
Applicant..Microsoft Corp.
Group Art Unit 2194
Examiner .... Anya, Charles E.
Attorney's Docket No. MS1-861USC1
Title: "Application Program Interface for Network Software Platform"

APPEAL BRIEF

To: Commissioner for Patents
        PO Box 1450
        Alexandria, Virginia 22313-1450

From: Lance Sadler (Tel. 509-324-9256x226; Fax 509-323-8979)
        Customer No. 22801

Pursuant to 37 C.F.R. §41.37, Applicant hereby submits an appeal brief

for application 10/087,027, filed February 28, 2002, within the requisite time from the

date of filing the Notice of Appeal. Accordingly, Applicant appeals to the Board of

Patent Appeals and Interferences seeking review of the Examiner's rejections.

Appeal Brief Items   **Page**

Application No. 10/087,027

(1)   Real Party in Interest

The real party in interest is Microsoft Corporation, the assignee of all right, title and interest in and to the subject invention.

(2)   Related Appeals and Interferences

Appellant is not aware of any other appeals, interferences, or judicial proceedings which will directly affect, be directly affected by, or otherwise have a bearing on the Board's decision to this pending appeal.

(3)   Status of Claims

Claims 1-41 stand rejected and are pending in the Application. Claims 1-41 are appealed. Some of these claims were previously amended. Claims 1-41 are set forth in the Appendix of Appealed Claims on page 42.

(4)   Status of Amendments

A Final Office Action (hereinafter "Office Action") was issued dated May 4, 2006 and no claims were amended responsive thereto.

A Notice of Appeal was filed on October 3, 2006.

(5)   Summary of Claimed Subject Matter

A concise explanation of each of the independent claims is included in this Summary section, including specific reference characters. These specific reference characters are examples of particular elements of the drawings for

certain embodiments of the claimed subject matter and the claims are not limited to solely the elements corresponding to these reference characters.

With regard to claim 1, a software architecture for a distributed computing system comprising an application configured to handle requests submitted by remote devices over a network (page 11, lines 6-25; See Fig.1); an application program interface to present functions used by the application to access network and computing resources of the distributed computing system (page 13, lines 6-25 – page 14, line 1-5; Fig. 1, 142); and a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 5, an application program interface embodied on one or more computer readable media, comprising a first group of services related to creating Web applications (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200); a second group of services related to constructing client applications (page 16, lines 13-18 – page 17, lines 9-13; Fig. 2, 202); a third group of services related to data and handling XML documents (page 16, lines 13-18 – page 17, lines 14-17; Fig. 2, 204); and a fourth group of services related to base class libraries (page 16, lines 13-19 – page 17, lines 17-21; Fig. 2, 206); and further comprising a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 11, a distributed computer software architecture, comprising one or more applications configured to be executed on one or more computing devices (page 12, lines 11-15), the one or more applications handling

requests submitted from remote computing devices (page 13, lines 6-25 – page 14, line 1-5; Fig. 1, 142); a networking platform to support the one or more applications (page 12, lines 16-23; Fig. 1, 110, 130, 134); an application programming interface to interface the one or more applications with the networking platform (page 13, lines 20-25 – page 14 line 1; Fig. 1, 142); and a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 18, a computer system including one or more microprocessors and one or more software programs, the one or more software programs utilizing an application program interface to request services from an operating system, the application program interface including separate commands to request services consisting of the following groups of services, a first group of services related to creating Web applications (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200); constructing Web services (page 18, lines 16-25 – page 19, lines 1-4; Fig. 3, 300); temporary caching resources (page 19, lines 5-12; Fig. 3, 308); performing initial configuration (page 19, lines 13-15; Fig. 3, 310); creating controls and Web pages (page 19, lines 16-25 – page 20, lines 1-7; Fig. 3, 312); enabling security in Web server applications (page 20, lines 8-11; Fig. 3, 318); accessing session state values (page 20, lines 12-15; Fig. 3, 318); a second group of services related to constructing client applications (page 16, lines 13-18 – page 17, lines 9-13; Fig. 2, 202); creating windowing graphical user interface environments (page 20, lines 19-25 – page 21, lines 1-8; Fig. 3, 322); enabling graphical functionality (page 21, lines 9-19; Fig. 3, 328); a third group of services

related to data and handling XML documents (page 16, lines 13-18 – page 17, lines 14-17; Fig. 2, 204); enabling management of data from multiple data sources (page 21, lines 23-25 – page 22, lines 1-18; Fig. 3, 340); second functions that enable XML processing (page 22, lines 19-25 – page 23, lines 1-5; Fig. 3, 350); a fourth group of services related to base class libraries (page 16, lines 13-19 – page 17, lines 17-21; Fig. 2, 206); defining various collections of objects (page 23, lines 10-12; Fig. 3, 360); accessing configuration settings and handling errors in configuration files (page 23, lines 13-16; Fig. 3, 362); debugging and tracing code execution (page 23, lines 17-21; Fig. 3, 364); customizing data according to cultural related information (page 23, lines 22-24 – page 24, lines 1-2; Fig. 3, 366); inputting and outputting of data (page 24, lines 3-9; Fig. 3, 368); enabling a programming interface to network protocols (page 24, lines 10-24; Fig. 3, 370); viewing loaded types, methods, and fields (page 25, lines 1-3; Fig. 3, 372); creating, storing and managing various culture-specific resources (page 25, lines 4-6; Fig. 3, 374); enabling system security and permissions (page 25, lines 7-9; Fig. 3, 376); installing and running services (page 25, lines 10-17; Fig. 3, 378); enabling character encoding (page 25, lines 18-23; Fig. 3, 380); enabling multi-threaded programming (page 25, lines 24-25 – page 26, lines 1-6; Fig. 3, 382); and facilitating runtime operations (page 26, lines 7-22; Fig. 3, 384).

With regard to claim 19, a system comprising means for exposing a first set of functions that enable browser/server communication (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200); means for exposing a second set of functions that enable drawing and construction of client applications (page 16, lines 13-18 – page 17, lines 9-13; Fig. 2, 202); means for exposing a third set of functions that

enable connectivity to data sources and XML functionality (page 16, lines 13-18 – page 17, lines 14-17; Fig. 2, 204); means for exposing a fourth set of functions that enable system and runtime functionality (page 16, lines 13-19 – page 17, lines 17-21; Fig. 2, 206); and means for translating Web applications written in different languages into an intermediate language supported by a common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 24, a method implemented at least in part by a computer comprising managing network and computing resources for a distributed computing system (page 11, lines 6-25; See Fig.1); exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system, the set of functions comprising first functions to facilitate browser/server communications (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200), second functions to facilitate construction of client applications (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200), third functions to facilitate connectivity to data sources and XML functionality (page 16, lines 13-18 – page 17, lines 14-17; Fig. 2, 204), and fourth functions to access system and runtime resources (page 16, lines 13-19 – page 17, lines 17-21; Fig. 2, 206); and providing a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 26, a method implemented at least in part by a computer comprising creating a first namespace with functions that enable browser/server communication (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200); creating a second namespace with functions that enable drawing and

<div align="center">7</div>

construction of client applications (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200); creating a third namespace with functions that enable connectivity to data sources and XML functionality (page 16, lines 13-18 – page 17, lines 14-17; Fig. 2, 204); creating a fourth namespace with functions that enable system and runtime functionality (page 16, lines 13-19 – page 17, lines 17-21; Fig. 2, 206); and providing a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 31, a method implemented at least in part by a computer comprising calling one or more first functions to facilitate browser/server communication (page 17, lines 7-9; Fig. 2, 200); calling one or more second functions to facilitate construction of client applications (page 17, lines 7-9; Fig. 2, 200); calling one or more third functions to facilitate connectivity to data sources and XML functionality (page 17, lines 14-17; Fig. 2, 204); calling one or more fourth functions to access system and runtime resources (page 17, lines 17-21; Fig. 2, 206); and using a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 36, a method implemented at least in part by a computer comprising receiving one or more calls to one or more first functions to facilitate browser/server communication (page 17, lines 7-9; Fig. 2, 200); receiving one or more calls to one or more second functions to facilitate construction of client applications (page 17, lines 7-9; Fig. 2, 200); receiving one or more calls to one or more third functions to facilitate connectivity to data

sources and XML functionality (page 17, lines 14-17; Fig. 2, 204); receiving one or more calls to one or more fourth functions to access system and runtime resources (page 17, lines 17-21; Fig. 2, 206); and using a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer (page 14, lines 1-5; Fig. 1, 144).

With regard to claim 41, a method implemented at least in part by a computer, for exposing resources using an application program interface comprising exposing a first group of services related to creating Web applications (page 16, lines 13-17 – page 17, lines 7-9; Fig. 2, 200), including, constructing Web services (page 18, lines 16-25 – page 19, lines 1-4; Fig. 3, 300); temporary caching resources (page 19, lines 5-12; Fig. 3, 308); performing initial configuration (page 19, lines 13-15; Fig. 3, 310); creating controls and web pages (page 19, lines 16-25 – page 20, lines 1-7; Fig. 3, 312); enabling security in Web server applications (page 20, lines 8-11; Fig. 3, 318); accessing session state values (page 20, lines 12-15; Fig. 3, 318); exposing a second group of services related to constructing client applications (page 16, lines 13-18 – page 17, lines 9-13; Fig. 2, 202), including, creating windowing graphical user interface environments (page 20, lines 19-25 – page 21, lines 1-8; Fig. 3, 322); enabling graphical functionality (page 21, lines 9-19; Fig. 3, 328); exposing a third group of services related to data and handling XML documents (page 16, lines 13-18 – page 17, lines 14-17; Fig. 2, 204), including, enabling management of data from multiple data sources (page 21, lines 23-25 – page 22, lines 1-18; Fig. 3, 340); second functions that enable XML processing (page 22, lines 19-25 – page 23,

Application No. 10/087,027

lines 1-5; Fig. 3, 350); exposing a fourth group of services related to base class libraries (page 16, lines 13-19 – page 17, lines 17-21; Fig. 2, 206), including, defining various collections of objects (page 23, lines 10-12; Fig. 3, 360); accessing configuration setting and handling errors in configuration files (page 23, lines 13-16; Fig. 3, 362); debugging and tracing code execution (page 23, lines 17-21; Fig. 3, 364); customizing data according to cultural related information (page 23, lines 22-24 – page 24, lines 1-2; Fig. 3, 366); inputting and outputting of data (page 24, lines 3-9; Fig. 3, 368); enabling a programming interface to network protocols (page 24, lines 10-24; Fig. 3, 370); viewing loaded types, methods, and fields (page 25, lines 1-3; Fig. 3, 372); creating, storing and managing various culture-specific resources (page 25, lines 4-6; Fig. 3, 374); enabling system security and permissions (page 25, lines 7-9; Fig. 3, 376); installing and running services (page 25, lines 10-17; Fig. 3, 378); enabling character encoding (page 25, lines 18-23; Fig. 3, 380); enabling multi-threaded programming (page 25, lines 24-25 – page 26, lines 1-6; Fig. 3, 382); and facilitating runtime operations (page 26, lines 7-22; Fig. 3, 384).

(6)   Grounds of Rejection to be Reviewed on Appeal

Claims 1-8, 10-16, 19-22, 24-29, 31-34, and 36-39 stand rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,792,605 B1 to Roberts et al. (hereinafter "Roberts") in view of U.S. Pub. No. 2002/0112078 A1 to Yach.

Claims 9, 17, 18, 23, 30, 35, 40, and 41 stand rejected under 35 U.S.C. § 103(a) as being obvious over Roberts in view of Yach as applied to claim 5, and further in view of U.S. Pat. No. 5,987,517 to Firth et al (hereinafter "Firth").

(7)   **Argument**

The arguments below are presented as follows. The first section will address the Office's rejections under § 103(a) over the combination of Roberts and Yach. The second section will then address the Office's rejections under § 103(a) over the combination of Roberts, Yach, and Firth.

  A. **The rejection under 35 U.S.C. §103(a) over the combination of Roberts and Yach does not establish a *prima facie* case of obviousness.**

   The §103 Standard

In making out a §103 rejection, the Federal Circuit has stated that when one or more reference or source of prior art is required in establishing obviousness, "it is necessary to ascertain whether the prior art *teachings* would appear to be sufficient to one of ordinary skill in the art to suggest making the claimed substitutions or other modification." *In re Fine,* 5 USPQ 2d, 1596, 1598 (Fed. Cir. 1988). That is, to make out a prima facie case of obviousness, the references must be examined to ascertain whether the combined *teachings* render the claimed subject matter obvious. *In re Wood*, 202 USPQ 171, 174 (C.C.P.A. 1979).

Moreover, there is a requirement that there must be some reason, suggestion, or motivation *from the prior art*, as a whole, for the person of ordinary skill to have combined or modified the references. *See, In re Geiger,* 2 USPQ 2d 1276, 1278 (Fed. Cir. 1987). It is impermissible to use the claimed invention as an instruction manual or "template" to piece together the teachings of the prior art so

that the claimed invention is rendered obvious. One cannot use hindsight reconstruction to pick and choose among isolated disclosures in the prior art to deprecate the claimed invention. *In re Fritch*, 23 USPQ 2d 1780, 1784 (Fed. Cir. 1992).

A factor cutting against a finding of motivation to combine or modify the prior art is when the prior art *teaches away* from the claimed combination. A reference is said to teach away when a person or ordinary skill, upon reading the reference, would be led in a direction divergent from the path that the applicant took. *In re Gurley*, 31 USPQ 2d 1130, 1131 (Fed. Cir 1994).

The need for specificity pervades this authority. See, e.g., *In re Kotzab*, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000) ("particular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed").

The Office's Attempted Combination of Roberts and Yach

In the discussion that follows, Applicant will describe the references to Roberts and Yach, and then provide reasons why the Office has not established a *prima facie* case of obviousness.

The Roberts Reference

Roberts is directed to a method and apparatus for accessing and using services and applications from a number of sources into a customized application.

Application No. 10/087,027

Roberts accomplishes this through an entity referred to as a web service. When a request for data or services is received, appropriate services are invoked by a web services engine using service drivers associated with each service. A web service application is then generated from a runtime model and is invoked to satisfy the request, by communicating as necessary with services in proper I/O formats.

Roberts discloses a system comprised of the web services directory and the web services engine. The web services directory is the database, while the web services engine is the processor that handles web service requests. A web service is a unit of computer processing that takes in a packet of request information and provides a packet of response information.

Roberts provides an instructive example in Column 5 starting at around line 53. As instructed by Roberts, a web service is represented externally as a URL that the web services engine processes to generate an HTTP response. While the web services engine is processing the web service, it employs a web service driver to perform the work. This driver can be self contained, or it can execute API's to interact with other systems. It can also request execution of web services that are maintained by other systems. Inside the web services architecture, a web service is comprised of two components. The first is a block of metadata that is stored in the web services directory; the second is a web service driver (implemented as a Java class) that runs in the context of the web service engine. The web service metadata block defines the schema of the web service's input and output interfaces, as well as configuration parameters that are used to configure the web service driver. In addition to having a metadata block that defines the "what" and "how" of a web service, a web service is represented by an entity in the web

services directory. This entity provides location information and access control privileges on the web service. As such, the directory provides the "who", "where", and "why" information associated with the web service.

Operation of Robert's web services engine is described in its Fig. 3.

The Yach Reference

Yach is directed to methods and systems for the browsing of documents without the need of a traditional document browsing application at a client device. The client device first transmits an information request to a host system. The host system retrieves the requested information from one or more information sources that store the information. A translation component receives the information from the host system and translates the information from a plurality of content types into a common virtual machine language program. The common virtual machine language program is then transmitted to the client device, which executes the virtual machine language program in order to display and interact with the information.

Yach describes one approach starting at paragraph [0010]. Specifically, an information source accessed via a TCP/IP network, using a standard information retrieval protocol like Hyper-text Transfer Protocol (HTTP), will provide information that will subsequently be translated into a common virtual machine language to be executed by a virtual machine (VM) operating at a client machine in place of traditional marked-up pages that must be rendered by a web browser. The traditional Internet browser or viewer is replaced with a virtual machine (VM)

Application No. 10/087,027

interpreter that is capable of dynamic program download and execution, and is also capable of providing advanced program behaviors and controls.

The Claims

Claim 1

Claim 1 recites a software architecture for a distributed computing system comprising [emphasis added]:

- an application configured to handle requests submitted by remote devices over a network;
- an application program interface to present functions used by the application to access network and computing resources of the distributed computing system; and
- *a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.*

In making out the rejection of this claim, the Office argues that its subject matter is rendered obvious in view of Roberts and Yach. Specifically, the Office argues that Roberts teaches a software architecture for a distributed system comprising: an application configured to handle requests submitted by remote devices over a network, and an application program interface to present functions used by the application to access network and computing resources of the distributed computing system. *See*, Office Action, page 2, paragraph 4.

Further, the Office argues that "Roberts is silent with reference to a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer." *See*, Office Action, page 3, paragraph 5. Applicant agrees.

The Office then further argues that "Yach teaches a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer", citing to Yach's translator 220, page 4, paragraph [0036].

The Office then argues that it would have been obvious "to combine the teachings of Yach and Roberts because the teaching of Yach would improve the system of Roberts by providing a method for eliminating the complexities between different runtime program environments and thus integrating their inter-relationship", citing to Yach, page 1, paragraphs [0004] and [0005]. *See,* Office Action, page 3, paragraph 7.

Applicant respectfully disagrees with the Office's reasoning and submits that the Office has failed to establish a *prima facie* case of obviousness for the following reasons: (1) the modification argued by the Office lacks the proper suggestion or motivation in either of the references; (2) the Office's combination does not appreciate the context of the references; (3) the combination of references would inevitably change an operating principle in Roberts; and (4) the Office has engaged in impermissible hindsight reconstruction in arguing for the combination of references.

Lack of Suggestion or Motivation to Support the Modification

First, Applicant submits that there is simply no suggestion or motivation in either of the references to support the modification argued by the Office.

Specifically, the Office states that it would have been obvious "to combine the teachings of Yach and Roberts because the teaching of Yach would improve

the system of Roberts by providing a method for eliminating the complexities between different runtime program environments and thus integrating their inter-relationship", citing to Yach, page 1, paragraphs [0004] and [0005]. *See,* Office Action, page 3, paragraph 7. Applicant respectfully reminds the Office that to establish a prima facie case of obviousness, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. See, e.g., *In re Jones,* 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). Simply citing to an excerpt from one reference and quoting verbatim from it does not rise to the level sufficient to make out a prima facie case of obviousness.

Furthermore, this motivation is lacking in the type of particularity that is required to make out a prima facie case of obviousness. That is, the Office's stated motivation is so general that it could serve as a motivation to modify Roberts in any manner whatsoever.

MPEP 2142 instructs, with regard to making out a rejection under §103 that "[t]he factual inquiry whether to combine references must be ***thorough and searching***." The need for specificity pervades this authority. See, e.g., *In re Kotzab,* 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000) ("particular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed").

Applicant respectfully submits that the Office's stated motivation is not a "particular finding" sufficient to support a prima facie case of obviousness.

The Office appears to simply be arguing that the combination would be obvious because it would make one reference better. However, simply "improving" a reference by making a particular combination does not rise to the level of establishing a *prima facie* case of obviousness.

The Office's Combination Does Not Appreciate The Context of the References

The Office's combination reflects a lack of appreciation for the context of the references.

The purpose of Yach, as instructed by Yach, "...relates to a software system and method for enabling the browsing of WWW content (or other forms of content) without the need for a traditional web browser application (or some other form of content interpretation application) operating at the client machine." *See Yach,* page 1, paragraph [0002]. Yach describes that "...there are a large number of problems [that] are growing and [] remain unsolved with the traditional browser. These problems include: (1) the growing incompatibility between content sent and rendered by browsers in the marketplace; (2) the growing divisions between 'scripting' languages used by browsers, for example Javascript and Visual Basic Script; (3) current browsers are becoming a tangled mess of runtime environments, including the browser runtime environment, the Javascript runtime environment, the Javascript runtime environment, the Java runtime environment and the Visual Basic runtime environment; (4) the inter-relationship between these runtime environments is creating a system too complex to debug and too large for small systems, particularly portable devices using wireless data networks; (5) *the growing size and complexity of web pages, specifically those*

Application No. 10/087,027

pages that include multimedia graphical and video content, is placing major strain on the Internet infrastructure and is impossible to support on limited bandwidth wireless devices; and (6) changing browsers, or versions, or updates, and the subsequent deployment to the marketplace necessitates frequent updates for each client machine using the browser." *See*, Yach, page 1, paragraph [0004] (emphasis added).

In solving the problems listed above, "the [disclosure] provides a *Virtual Machine (VM)* at the client machine in place of the much larger traditional web browser applications...." *See,* page 1, paragraph [0006]. In addition, Yach discloses that "[t]he invention solves problems with every growing web page content by using a programmatic language (such as virtual machine language) to replace a page rendering language (such as HTML). *See,* page 1, paragraph [0008].

Roberts, on the other hand, is directed to a method and apparatus for accessing and using services and applications from a number of sources into a customized application. Roberts accomplishes this through an entity referred to as a web service. When a request for data or services is received, appropriate services are invoked by a web services engine using service drivers associated with each service. A web service application is then generated from a runtime model and is invoked to satisfy the request, by communicating as necessary with services in proper I/O formats.

Thus, the context of Yach pertains to replacing traditional web browsers with its virtual machine that resides at the client. Roberts, on the other hand, pertains to consumption of web services through its particularly described

architecture. Nowhere in Roberts is there any mention or hint of a problem that could be solved by Yach's use of a programmatic language to replace a page rendering language. In many senses, these references are operating in very different problem spaces and hence, when viewed in this context, the Office's combination does not appear to make technical sense.

The Combination of the References Would Change an Operating Principle of Roberts

Further, the combination of Roberts and Yach is improper since it would inevitably change an operating principle of Roberts.

If the proposed modification or combination of the prior art would change the principle of operation of the prior art invention being modified, then the teachings of the references are not sufficient to render the claims prima facie obvious. *In re Ratti*, 270 F.2d 810, 123 USPQ 349 (CCPA 1959) (Claims were directed to an oil seal comprising a bore engaging portion with outwardly biased resilient spring fingers inserted in a resilient sealing member. The primary reference relied upon in a rejection based on a combination of references disclosed an oil seal wherein the bore engaging portion was reinforced by a cylindrical sheet metal casing. Patentee taught the device required rigidity for operation, whereas the claimed invention required resiliency. The court reversed the rejection holding the "suggested combination of references would require a substantial reconstruction and redesign of the elements shown in [the primary reference] as well as a change in the basic principle under which the [primary reference] construction was designed to operate." *Id.* at 813.)

The Office cites the following excerpts of Yach in making out its rejection:

20

[0004] Web browsers are well known in the art. Products such as Microsoft's Internet Explorer.RTM. and Netscape's Communicator.RTM. dominate the market for browsing WWW content. Those skilled in the art of Internet browsing will appreciate that a large number of problems are growing and continue to remain unsolved with these traditional browser applications. These problems include: (1) the growing incompatibility between content sent and rendered by browsers in the marketplace; (2) the growing divisions between `scripting` languages used by browsers, for example, Javascript and Visual Basic Script; (3) current browsers are becoming a tangled mess of runtime environments, including the browser runtime environment, the Javascript runtime environment, the Java runtime environment and the Visual Basic runtime environment; (4) the inter-relationship between these runtime environments is creating a system too complex to debug and too large for small systems, particularly portable devices using wireless data networks; (5) the growing size and complexity of web pages, specifically those pages that include multimedia graphical and video content, is placing major strain on the Internet infrastructure and is impossible to support on limited-bandwidth wireless devices; and (6) changing browsers, or versions, or updates, and the subsequent deployment to the marketplace necessitates frequent updates for each client machine using the browser. This last problem is compounded with mobile devices, such as cell phones, pagers, and handheld devices, which typically cannot be upgraded over-the-air because of security concerns and rollover complexities.

[0005] A system and method of browsing documents is provided that does not require a traditional document browsing application at a client device. In order to achieve browsing without a browsing application, the client device first transmits an information request to a host system. The host system retrieves the requested information from one or more information sources that store the information. A translation component receives the information from the host system and translates the information from a plurality of content types into a common virtual machine language program. The common virtual machine language program is then transmitted to the client device, which executes the virtual machine language program in order to display and interact with the information.

This passage describes the notion of a system that does not require a traditional document browser application. That is, in solving the problem of

Application No. 10/087,027

browser complexity and deployment, the ***invention provides a Virtual Machine (VM) at the client machine in place of the much larger traditional web browser applications***.

Roberts, on the other hand, "provides a method and apparatus for accessing and using service and applications from a number of sources into a customized application. The present invention accomplishes this through an entity referred to as a web service...the web services application provides responses in the form of ***HTML that can be used to generate pages to a browser.*** *See,* col. 3, lines 22-26 and lines 36-38.

Making the combination suggested by the Office would appear to require Roberts to change its entire architecture in order to support Yach's described functionality. Specifically, Roberts' architecture would have to be modified to receive and process Yach's common virtual machine language program. This would entail, among other things, Roberts replacing any generated markup pages that were formerly rendered by its browser with Yach's common virtual machine language that is to be executed by Yach's virtual machine.

Insofar as Roberts instructs that its web services application provides responses in the form of HTML that can be used to generate pages to a browser, and insofar as Roberts does not suggest any alternatives that would lead one to consider solutions proposed by Yach, it appears inevitable that the Office's combination would impermissibly change an operating principle of Roberts.

Accordingly, for this additional reason, the Office has failed to establish a *prima facie* case of obviousness.

The Office has Engaged in Impermissible Hindsight Reconstruction

Further, the stated motivation to combine the cited references is insufficient because the Office has engaged in improper hindsight reconstruction in arguing this combination of references. As noted above, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Here, Roberts and Yach contain no such suggestion or motivation. Furthermore, the Office bears the burden of explaining "why the combination of the teachings is proper." MPEP §2142. Here, the Office's only attempt at such an explanation is to state that an artisan would have been motivated to modify Roberts with the teachings of Yach in order to "improve the system of Roberts by providing a method for eliminating the complexities between different runtime program environments and thus integrating their inter-relationship." *See,* Office Action, page 3, paragraph 7. However, this reasoning is misplaced. It does not appear that Roberts expresses any concern about or need to mitigate the effects of complexities between different runtime program environments. To be sure, this motivation comes directly from Yach which, when taken in the context of Roberts, does not appear to make sense.

Because the Office does not explain how or why Roberts suffers from complexities associated with different runtime program environments such that Yach would provide mitigation therefore, and because Roberts does not teach the use of a virtual machine-type approach at all, the combination of these references is improper.

Application No. 10/087,027

In conclusion, and at least for the reasons discussed above, the Office has failed to establish a *prima facie* case of obviousness with respect to this claim and this claim is allowable.

Claims 2-4

Claims 2-4 are allowable as depending from an allowable base claim.

Claim 5

Claim 5 recites an application program interface embodied on one or more computer readable media, comprising (emphasis added):

- a first group of services related to creating Web applications;
- a second group of services related to constructing client applications;
- a third group of services related to data and handling XML documents; and
- a fourth group of services related to base class libraries; *and further comprising:*
    - *a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.*

In making out the rejection of this claim, the Office argues that Roberts teaches the software architecture as recited in claim 4, wherein the application program interface comprises: a first group of services related to creating Web applications (Col. 7 Ln. 50-67, Col. 9 Ln. 27-35); a second group of services related to constructing client applications (Col. 14 Ln. 30-46); a third group of

services related to data and handling XML documents (Col. 10 Ln. 1-9, Ln. 59-67); and a fourth group of services related to base class libraries (Col. 6 Ln. 7-9, Col. 8 Ln. 29-38, Ln. 64-67). *See*, Office Action, page 4, paragraph 10.

For all of the reasons set forth above with regard to the combinability of Roberts and Yach, the Office has not established a *prima facie* case of obviousness. In addition, the specific excerpts of Roberts cited by the Office do not disclose "groups" of services as that term is used and specifically recited in Applicant's claims.

Accordingly, for at least these reasons, this claim is allowable.

Claims 6-10

Claims 6-10 depend from claim 5 and are allowable as depending from an allowable base claim.

Claim 11

Claim 11 recites a distributed computer software architecture, comprising (emphasis added):

- one or more applications configured to be executed on one or more computing devices, the *one or more* applications handling requests submitted from remote computing devices;
- a networking platform to support the one or more applications;
- an application programming interface to interface the one or more applications; and
- *a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.*

In making out the rejection of this claim, the Office argues that its subject matter is rendered obvious in view of Roberts and Yach and relies primarily on its arguments with regard to claim 1. For all of the reasons set forth above with regard to claim 1, the Office has failed to establish a *prima facie* case of obviousness. Accordingly, this claim is allowable.

Claims 12-17

Claims 12-17 depend from claim 11 and are allowable as depending from an allowable base claim. In addition, Applicant submits that the Office has failed to establish a *prima facie* case of obviousness in rejecting claims 12-17 by failing to establish that each and every element, in combination with claim 11, is taught or suggested by the reference.

Claim 19

Claim 19 recites a system comprising [emphasis added]:

- means for exposing a first set of functions that enable browser / server communication;
- means for exposing a second set of functions that enable drawing and construction of client applications;
- means for exposing a third set of functions that enable connectivity to data sources and XML functionality; and
- means for exposing a fourth set of functions that enable system and runtime functionality; and
- ***means for translating Web applications written in different languages into an intermediate language supported by a common runtime layer.***

In making out the rejection of this claim, the Office argues that "Roberts teaches the system comprising: means for exposing a set of functions that enable browser/server communication; means for exposing a second set of functions that enable drawing and construction of client applications (Col. 14 Ln. 30-46); means for exposing a third set of functions that enable connectivity to data sources and XML functionality (Col. 5 Ln. 25-37, Col. 10 Ln. 1-9, Ln. 59-67); and means for exposing a fourth set of functions that enable system and runtime functionality (Col. 8 Ln. 22-28)." *See,* Office Action, page 6, paragraph 19.

The portion of Roberts relied on by the Office to reject claim 19 is:

> **Col. 5, Lines 25-37:** A web service is a unit of computer processing that takes in a packet of request information and provides a packet of response information. The protocol for a web service in one embodiment is HTTP. However, the protocol can be any suitable protocol or any future protocol. HTTP is used as an example for one embodiment. The request and response data format is XML in one embodiment, but it could be any other suitable format or combination of formats, including, but not limited to, HTML, MIME encoded types such as gif, or any other format. A web service is represented by a URL, along with a schema that defines the range of input and output data formats.
>
> **Col. 8, Lines 22-28:** These parameters may be obtained at runtime from a URL.
>
> Examples of service parameters associated with commonly used service driver classes are: the e-mail server name for the e-mail driver class, the URL to be called by the HTTP driver class, the runtime model content for a model runner driver class.
>
> **Col. 10, Lines 1-9:** This is shown in FIGS. 2A and 2B. Referring first to FIG. 2A, an example of remote procedure call/application programmer interface (RPC/API) accessible applications is shown. The system interacts with the service via.XML in/out interface. Service driver A converts XML requests to satisfy the appropriate API of the application or message broker 201 being accessed and forwards the reconfigured request. The application returns a response, which the Service Driver A converts into an XML response.
>
> **Col. 14, Lines 30-46:** The Pages entity of a runtime model contains a set of individual page entities. A runtime model can have any number of

pages. Pages serve as a vehicle for both content and UI that is suitably formatted for display in a browser.

Once the model runner service driver has produced a WSA, it executes a block of functionality in the WSA, that ultimately sends back a response to the requester. Many times, the response is in the form of an page, suitable for display in a browser.

Since a page as just another type of parameterized XML data in the WSA session, a WSA can utilize a page as output to a service request, or as input to another web service. This means that a WSA can send pages as inputs to a variety of web services. For example, a WSA could send a page to an email web service as the content of a message.

The portion of Roberts relied on by the Office does not teach or suggest "means for translating Web applications written in different languages into an intermediate language supported by a common runtime layer." If it is the Office's intention to rely on Yach for this feature (which is unclear from the Office Action), then Applicant submits for reasons analogous to those argued with regard to claim 1, that the Office has failed to establish a *prima facie* case of obviousness.

Accordingly, for at least this reason, this claim is allowable.

Claims 20-22

Claims 20-22 depend from claim 19 and are allowable as depending from an allowable base claim. In addition, Applicant submits that the Office has failed to establish a *prima facie* case of obviousness in rejecting claims 20-22 by failing to establish that each and every element, in combination with claim 19, is taught or suggested by the reference.

Claim 24 recites a method implemented at least in part by a computer, comprising (emphasis added):

- managing network and computing resource for a distributed computing system;
- exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system, the set of functions comprising first functions to facilitate browser/server communication, second functions to facilitate construction of client applications, third functions to facilitate connectivity to data sources and XML functionality, and fourth functions to access system and runtime resources; and
- *providing a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.*

In making out the rejection of this claim, the Office relies on its argument with regard to claim 19. Applicant submits for reasons analogous to those argued above with regard to claim 19, that the Office has failed to establish a *prima facie* case of obviousness.

Accordingly, for at least this reason, this claim is allowable.

Claim 25

Claim 25 depends from claim 24 and is allowable as depending from an allowable base claim. In addition, Applicant submits that the Office has failed to establish a *prima facie* case of obviousness in rejecting claim 25 by failing to establish that each and every element, in combination with claim 24, is taught or suggested by the reference.

<u>Claim 26</u>

Claim 26 recites a method implemented at least in part by a computer, comprising (emphasis added):

- creating a first namespace with functions that enable browser/server communication;
- creating a second namespace with functions that enable drawing and construction of client applications;
- creating a third namespace with functions that enable connectivity to data sources and XML functionality; and
- ***providing a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.***

In making out the rejection of this claim, the Office simply cites to the rejection that it made out for claim 19. Applicant submits for reasons analogous to those argued above with regard to claim 19, that the Office has failed to establish a *prima facie* case of obviousness. In addition, this claim includes the notion of creating namespaces as recited above. Claim 19, however, does not specifically call out the notion of namespaces (which does not mean that it does not cover the notion of namespaces). Rather, Applicant merely points this out because the Office has not specifically applied the references and pointed out where the notion of namespaces, as recited in this claim, is disclosed or suggested.

Accordingly, for at least these reasons, this claim is allowable.

<u>Claims 27-29</u>

Claims 27-29 depend from claim 26 and are allowable as depending from an allowable base claim. In addition, Applicant submits that the Office has failed to establish a *prima facie* case of obviousness in rejecting claims 27-29 by failing to establish that each and every element, in combination with claim 26, is taught or suggested by the reference.

<u>Claim 31</u>

Claim 31 recites a method implemented at least in part by a computer, comprising (emphasis added):

- calling one or more first functions to facilitate browser/server communication;
- calling one or more second functions to facilitate construction of client applications;
- calling one or more third functions to facilitate connectivity to data sources and XML functionality;
- calling one or more fourth functions to access system and runtime resources; and
- ***using a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.***

In making out the rejection of this claim, the Office simply refers to its rejection of claim 24. For reasons analogous to those argued with regard to claim 24, the Office has failed to establish a *prima facie* case of obviousness.

Accordingly, for at least this reason, this claim is allowable.

<u>Claims 32-34</u>

Claims 32-34 depend from claim 31 and are allowable as depending from an allowable base claim. In addition, Applicant submits that the Office has failed to establish a *prima facie* case of obviousness in rejecting claims 32-34 by failing to establish that each and every element, in combination with claim 31, is taught or suggested by the reference.

<u>Claim 36</u>

Claim 36 recites a method implemented at least in part by a computer, comprising:

- receiving one or more calls to one or more first functions to facilitate browser/server communications;
- receiving one or more calls to one or more second functions to facilitate connectivity to data sources and XML functionality;
- receiving one or more calls to one or more fourth functions to access system and runtime resources; and
- *using a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.*

In making out the rejection of this claim, the Office refers to its argument with regard to claim 19. For reasons analogous to those argued with regard to claim 19, the Office has failed to establish a prima facie case of obviousness.

Accordingly, for at least this reason, this claim is allowable.

<u>Claims 37-39</u>

Claims 37-39 depend from claim 36 and are allowable as depending from an allowable base claim. In addition, Applicant submits that the Office has failed

Application No. 10/087,027

to establish a *prima facie* case of obviousness in rejecting claims 37-39 by failing to establish that each and every element, in combination with claim 36, is taught or suggested by the reference.

B. **The rejection under 35 U.S.C. §103(a) over the combination of Roberts, Yach, and Firth does not establish a *prima facie* case of obviousness.**

In the discussion that follows, Applicant will describe the reference to Firth and then provide reasons why the Office has not established a *prima facie* case of obviousness.

The Firth Reference

Firth is directed to creating computer network applications. More specifically, it relates to creating simplified computer network applications by using a library of reentrant network functions which allow an application to reduce the source code required to interact with a computer network such as the Internet.

The Office Has Failed to Establish a Prima Facie Case of Obviousness With Regard to the Combination of Roberts and Firth

Applicant submits that the Office has failed to establish a *prima facie* case of obviousness with regard to the combination of Roberts and Firth for the following reasons: (1) the modification argued by the Office lacks the proper suggestion or motivation in either of the references; (2) the Office has engaged in impermissible hindsight reconstruction in arguing for the combination of references.

Application No. 10/087,027

Lack of Suggestion or Motivation to Support the Modification

First, Applicant submits that there is simply no suggestion or motivation in either of the references to support the modification argued by the Office.

Specifically, the Office states that it would have been obvious "to combine the teachings of Firth and Roberts because the teaching of Firth would improve the system of Roberts by creating computer network applications by using a library of reentrant network functions which allow an application to reduce the source code required to interact with a computer network such as the internet." *See,* Office Action, page 8-9, paragraph 33.

Applicant respectfully reminds the Office that to establish a prima facie case of obviousness, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. See, e.g., *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992). Here, while the motivation come verbatim out of Firth, it appears to have no context with regard to Roberts. Specifically, Roberts has expressed no problem that would be served by the technology described in Firth.

Moreover, this motivation is lacking in the type of particularity that is required to make out a prima facie case of obviousness. That is, the Office's stated motivation is so general that it could serve as a motivation to modify Roberts in any manner whatsoever.

MPEP 2142 instructs, with regard to making out a rejection under §103 that "[t]he factual inquiry whether to combine references must be ***thorough and searching***." The need for specificity pervades this authority. See, e.g., *In re*

Kotzab, 217 F.3d 1365, 1371, 55 USPQ2d 1313, 1317 (Fed. Cir. 2000) ("particular findings must be made as to the reason the skilled artisan, with no knowledge of the claimed invention, would have selected these components for combination in the manner claimed").

Applicant respectfully submits that the Office's stated motivation is not a "particular finding" sufficient to support a prima facie case of obviousness. Rather, the Office's "particular finding" is a clipped excerpt from Firth that describes, in Firth's context, the notion of using a library of reentrant network functions, without any regard for Robert's context. Applicant respectfully submits that in order to support a *prima facie* case of obviousness, the "particular findings" that must be made *must* be more than simply a clipped excerpt of one of the references that describes advantages of that particular reference's technology. To hold otherwise is to complete eviscerate the §103 standard.

The Office has Engaged in Impermissible Hindsight Reconstruction

Further, the stated motivation to combine the cited references is insufficient because the Office has engaged in improper hindsight reconstruction in arguing for this combination of references. As noted above, there must be some suggestion or motivation, either in the references themselves or in the knowledge generally available to one of ordinary skill in the art, to modify the reference or to combine reference teachings. Here, Roberts and Firth contain no such suggestion or motivation. Furthermore, the Office bears the burden of explaining "why the combination of the teachings is proper." MPEP §2142. Here, the Office's only attempt at such an explanation is to state that an artisan would have been

motivated to modify Roberts with the teachings of Firth in order to "improve the system of Roberts by creating computer network applications by using a library of reentrant network functions which allow an application to reduce the source code required to interact with a computer network such as the internet." *See,* Office Action, page 8-9, paragraph 33. However, this reasoning is misplaced.

Because the Office does not explain how Roberts suggests the need to reduce the source code required to interact with a computer network, the combination of the references is improper.

In conclusion, and at least for the reasons discussed above, the Office has failed to establish a *prima facie* case of obviousness with respect to this claim and this claim is allowable.

Claim 9

Claim 9 recites an application program interface as recited in ***claim 5***, wherein the fourth group of services comprises:

- first functions that enable definitions of various collections of objects;
- second functions that enable programmatic access to configuration settings and handling of errors in configurations files;
- third functions that enable application debugging and code execution tracing;
- fourth functions that enable customization of data according to cultural related information;
- fifth functions that enable input/output of data;
- sixth functions that enable a programming interface to network protocols;
- seventh functions that enable a managed view of types, methods, and fields;
- eighth functions that enable creation, storage and management of various culture-specific resources;

Application No. 10/087,027

- ninth functions that enable system security and permissions;
- tenth functions that enable installation and running of services;
- eleventh functions that enable character encoding;
- twelfth functions that enable multi-threaded programming; and
- thirteenth functions that facilitate runtime operations.

In making out the rejection of this claim, the Office argues that its subject matter is rendered obvious in view of Roberts, Yach and Firth. Specifically, the Office argues that "Roberts teaches the application program interface as recited in claim 5." *See,* Office Action, pages 7-8, paragraph 30.

The Office argues that "Roberts is silent with reference to second functions that enable programmatic access to configuration settings and handling or error in configuration files; third functions that enable application debugging and code execution tracing; fourth functions that enable customization of data according to cultural related information; seventh functions that enable a managed view of types, methods, and fields; eighth functions that enable culture-specific resources and twelfth functions that enable multi-threaded programming." *See,* Office Action, page 8, paragraph 31. The Office further argues that "Firth teaches second functions that enable programmatic access to configuration settings and handling of errors in configuration files; third functions that enable application debugging and code execution tracing; fourth functions that enable customization of data according to cultural related information; eighth functions that enable culture specific resources; seventh functions that enable a managed view of types, methods, and fields; twelfth functions that enable multi-threaded programming." *See,* Office Action, page 8, paragraph 32 (Cites omitted). The Office concludes its argument with respect to claim 9 by arguing that it would have been obvious "to combine the teachings of Firth and Roberts because the teaching of Firth would

Application No. 10/087,027

improve the system of Roberts by creating computer network applications by using a library of reentrant network functions which allow an application to reduce the source code required to interact with a computer network such as the internet", citing to *Firth*, Col. 1, Ln. 9-14]. *See,* Office Action, page 9, paragraph 33.

For all of the reasons set forth with regard to claim 5, this claim is allowable. In addition, this claim is allowable for the additional reasons mentioned above.

Claim 17

Claim 17 depends from claim 11. Accordingly, this claim is allowable as depending from an allowable base claim. Further, in making out the rejection of this claim, the Office relies on its arguments with regard to claim 9. For reasons analogous to those argued with regard to claim 9, this claim is allowable.

Claim 18

Claim 18 recites a computer system including one or more microprocessors and one or more software programs, the one or more software programs utilizing an application program interface to request services from an operating system, the application program interface including separate commands to request services consisting of the following groups of services:

- a first group of services related to creating Web applications:
  - o Constructing Web services;
  - o Temporary caching resources;
  - o Performing initial configuration;
  - o Creating controls and Web pages;
  - o Enabling security in Web server applications;
  - o Accessing session state values;

- a second group of services related to constructing client applications:
  - creating windowing graphical user interface environments;
  - enabling graphical functionality
- a third group of services related to data and handling XML documents:
  - enabling management of data from multiple data sources;
  - second functions that enable XML processing.
- a fourth group of services related to base class libraries:
  - defining various collections of objects;
  - accessing configuration settings and handling errors in configuration files;
  - debugging and tracing code execution;
  - customizing data according to cultural related information;
  - inputting and outputting of data;
  - enabling a programming interface to network protocols;
  - viewing loaded types, methods, and field;
  - creating storing and managing various culture-specific resources;
  - enabling system security and permissions;
  - installing and running services;
  - enabling character encoding;
  - enabling multi-threaded programming; and
  - facilitating runtime operations.

In making out the rejection of claim 18, the Office simply cites to its rejection of claim 9. Applicant respectfully submits that this claim includes subject matter that is not specifically recited in claim 9. To this extent, this claim has not been properly examined. Further, to the extent that this claim includes subject matter that is analogous to the subject matter recited in claim 9, this claim is allowable for analogous reasons.

Claims 23, 30, and 35

In making out the rejection of claim 23, 30, and 35 the Office simply cites to its rejection of claim 9. To the extent that these claims include subject matter

that is the same as or analogous to the subject matter appearing in claim 9, these claims are allowable for the same reason.

Claim 41

Claim 41 recites a method for implemented at least in part by a computer, for exposing resources using an application program interface comprising:

- exposing a first group of services related to creating Web applications, including:
  - constructing Web services;
  - temporary caching resources;
  - performing initial configuration;
  - creating controls and Web pages;
  - enabling security in Web server applications;
  - accessing session state values;
- exposing a second group of services related to constructing client applications, including:
  - creating windowing graphical user interface environments;
  - enabling graphical functionality;
- exposing a third group of services related to data and handling XML documents, including:
  - enabling management of data from multiple data sources;
  - second functions that enable XML processing.
- exposing a fourth group of services related to base class libraries, including:
  - defining various collections of objects;
  - accessing configuration settings and handling errors in configuration files;
  - debugging and tracing code execution;
  - customizing data according to cultural related information;
  - inputting and outputting of data;
  - enabling a programming interface to network protocols;
  - viewing loaded types, methods, and fields;

Application No. 10/087,027

- creating, storing and managing various culture-specific resources;
- enabling system security and permissions;
- installing and running services;
- enabling character encoding;
- enabling multi-threaded programming; and
- facilitating runtime operations.

In making out the rejection of claim 41, the Office simply cites to its rejection of claim 9. Applicant respectfully submits that this claim includes subject matter that is not specifically recited in claim 9. To this extent, this claim has not been properly examined. Further, to the extent that this claim includes subject matter that is analogous to the subject matter recited in claim 9, this claim is allowable for analogous reasons.
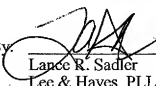
Conclusion

For the reasons mentioned above, the Office has failed to establish a *prima facie* case of obviousness. Accordingly, Applicant respectfully requests that the rejections be overturned and that the pending claims be allowed to issue.

Respectfully Submitted,

Dated: _12/1/06_

By: _____
Lance R. Sadler
Lee & Hayes, PLLC
Reg. No. 38,605
(509) 324-9256 ext. 226

(8)   Appendix of Appealed Claims

1.   (Previously Presented) A software architecture for a distributed computing system comprising:

an application configured to handle requests submitted by remote devices over a network;

an application program interface to present functions used by the application to access network and computing resources of the distributed computing system; and

a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.

2.   (Previously Presented) The software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one client device.

3.   (Previously Presented) The software architecture as recited in claim 1, wherein the distributed computing system comprises client devices and server devices that handle requests from the client devices, the remote devices comprising at least one server device that is configured as a Web server.

4.   (Previously Presented) The software architecture as recited in claim 1, wherein the application program interface comprises:

a first group of services related to creating Web applications;

a second group of services related to constructing client applications;

a third group of services related to data and handling XML documents; and

a fourth group of services related to base class libraries.

5. (Previously Presented) An application program interface embodied on one or more computer readable media, comprising:

a first group of services related to creating Web applications;

a second group of services related to constructing client applications;

a third group of services related to data and handling XML documents; and

a fourth group of services related to base class libraries; and further comprising:

a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.

6. (Previously Presented) The application program interface as recited in claim 5, wherein the first group of services comprises:

first functions that enable construction and use of Web services;

second functions that enable temporary caching of frequently used resources;

third functions that enable initial configuration;

fourth functions that enable creation of controls and Web pages;

fifth functions that enable security in Web server applications; and

sixth functions that enable access to session state values.

7. (Previously Presented) The application program interface as recited in claim 5, wherein the second group of services comprises:

first functions that enable creation of windowing graphical user interface environments; and

second functions that enable graphical functionality.

8. (Previously Presented) The application program interface as recited in claim 5, wherein the third group of services comprises:

first functions that enable management of data from multiple data sources; and

second functions that enable XML processing.

9. (Previously Presented) The application program interface as recited in claim 5, wherein the fourth group of services comprises:

first functions that enable definitions of various collections of objects;

second functions that enable programmatic access to configuration settings and handling of errors in configuration files;

third functions that enable application debugging and code execution tracing;

fourth functions that enable customization of data according to cultural related information;

fifth functions that enable input/output of data;

sixth functions that enable a programming interface to network protocols;

seventh functions that enable a managed view of types, methods, and fields;

eighth functions that enable creation, storage and management of various culture-specific resources;

ninth functions that enable system security and permissions;

tenth functions that enable installation and running of services;

eleventh functions that enable character encoding;

twelfth functions that enable multi-threaded programming; and

thirteenth functions that facilitate runtime operations.

10. (Original) A network software architecture comprising the application program interface as recited in claim 5.

11.   (Previously Presented) A distributed computer software architecture, comprising:

one or more applications configured to be executed on one or more computing devices, the one or more applications handling requests submitted from remote computing devices;

a networking platform to support the one or more applications;

an application programming interface to interface the one or more applications with the networking platform; and

a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.

12.   (Previously Presented) The distributed computer software architecture as recited in claim 11, further comprising a remote application configured to be executed on one of the remote computing devices, the remote application using the application programming interface to access the networking platform.

13.   (Previously Presented) The distributed computer software architecture as recited in claim 11, wherein the application programming interface comprises:

a first group of services related to creating Web applications;

a second group of services related to constructing client applications;

a third group of services related to data and handling XML documents; and

a fourth group of services related to base class libraries.

14.   (Previously Presented) The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

45

first functions that enable construction and use of Web services;

second functions that enable temporary caching of frequently used resources;

third functions that enable initial configuration;

fourth functions that enable creation of controls and Web pages;

fifth functions that enable security in Web server applications; and

sixth functions that enable access to session state values.

15.   (Previously Presented) The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

first functions that enable creation of windowing graphical user interface environments; and

second functions that enable graphical functionality.

16.   (Previously Presented) The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

first functions that enable management of data from multiple data sources; and

second functions that enable XML processing.

17.   (Previously Presented) The distributed computer software architecture as recited in claim 11, wherein the application programming interface exposes multiple functions comprising:

first functions that enable definitions of various collections of objects;

second functions that enable programmatic access to configuration settings and handling of errors in configuration files;

third functions that enable application debugging and code execution tracing;

fourth functions that enable customization of data according to cultural related information;

fifth functions that enable input/output of data;

sixth functions that enable a programming interface to network protocols;

seventh functions that enable a managed view of loaded types, methods, and fields;

eighth functions that enable creation, storage and management of various culture-specific resources;

ninth functions that enable system security and permissions;

tenth functions that enable installation and running of services;

eleventh functions that enable character encoding;

twelfth functions that enable multi-threaded programming; and

thirteenth functions that facilitate runtime operations.

18.   (Original) A computer system including one or more microprocessors and one or more software programs, the one or more software programs utilizing an application program interface to request services from an operating system, the application program interface including separate commands to request services consisting of the following groups of services:

A. a first group of services related to creating Web applications:

constructing Web services;

temporary caching resources;

performing initial configuration;

creating controls and Web pages;

enabling security in Web server applications;

accessing session state values;

B. a second group of services related to constructing client applications:

creating windowing graphical user interface environments;

enabling graphical functionality;

C. a third group of services related to data and handling XML documents:

enabling management of data from multiple data sources;

second functions that enable XML processing.

D. a fourth group of services related to base class libraries:

defining various collections of objects;

accessing configuration settings and handling errors in configuration files;

debugging and tracing code execution;

customizing data according to cultural related information;

inputting and outputting of data;

enabling a programming interface to network protocols;

viewing loaded types, methods, and fields;

creating, storing and managing various culture-specific resources;

enabling system security and permissions;

installing and running services;

enabling character encoding;

enabling multi-threaded programming; and

facilitating runtime operations.

19.   (Previously Presented) A system comprising:

means for exposing a first set of functions that enable browser/server communication;

means for exposing a second set of functions that enable drawing and construction of client applications;

means for exposing a third set of functions that enable connectivity to data sources and XML functionality; and

means for exposing a fourth set of functions that enable system and runtime functionality; and

means for translating Web applications written in different languages into an intermediate language supported by a common runtime layer.

20. (Previously Presented) The system as recited in claim 19, wherein the first set of functions comprises:

first functions that enable construction and use of Web services;

second functions that enable temporary caching of frequently used resources;

third functions that enable initial configuration;

fourth functions that enable creation of controls and Web pages;

fifth functions that enable security in Web server applications; and

sixth functions that enable access to session state values.

21. (Previously Presented) The system as recited in claim 19, wherein the second set of functions comprises:

first functions that enable creation of windowing graphical user interface environments; and

second functions that enable graphical functionality.

22. (Previously Presented) The system as recited in claim 19, wherein the third set of functions comprises:

first functions that enable management of data from multiple data sources; and

second functions that enable XML processing.

23. (Previously Presented) The system as recited in claim 19, wherein the fourth set of functions comprises:

first functions that enable definitions of various collections of objects;

second functions that enable programmatic access to configuration settings and handling of errors in configuration files;

third functions that enable application debugging and code execution tracing;

fourth functions that enable customization of data according to cultural related information;

fifth functions that enable input/output of data;

sixth functions that enable a programming interface to network protocols;

seventh functions that enable a managed view of loaded types, methods, and fields;

eighth functions that enable creation, storage and management of various culture-specific resources;

ninth functions that enable system security and permissions;

tenth functions that enable installation and running of services;

eleventh functions that enable character encoding;

twelfth functions that enable multi-threaded programming; and

thirteenth functions that facilitate runtime operations.

24. (Previously Presented) A method implemented at least in part by a computer, comprising:

managing network and computing resources for a distributed computing system;

exposing a set of functions that enable developers to access the network and computing resources of the distributed computing system, the set of functions comprising first functions to facilitate browser/server communication, second functions to facilitate construction of client applications, third functions to facilitate connectivity to data sources and XML functionality, and fourth functions to access system and runtime resources; and

providing a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.

25.   (Previously Presented) The method as recited in claim 24, further comprising receiving a request from a remote computing device, the request containing a call to at least one of the first, second, third, and fourth functions.

26.   (Previously Presented) A method implemented at least in part by a computer, comprising:

creating a first namespace with functions that enable browser/server communication;

creating a second namespace with functions that enable drawing and construction of client applications;

creating a third namespace with functions that enable connectivity to data sources and XML functionality;

creating a fourth namespace with functions that enable system and runtime functionality; and

providing a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.

27.   (Previously Presented) The method as recited in claim 26, wherein the first namespace defines classes that facilitate:

construction and use of Web services;

temporary caching of resources;

initial configuration;

creation of controls and Web pages;

security in Web server applications; and

Application No. 10/087,027

access to session state values.

28.   (Previously Presented) The method as recited in claim 26, wherein the second namespace defines classes that facilitate:
creation of windowing graphical user interface environments; and
graphical functionality.

29.   (Previously Presented) The method as recited in claim 26, wherein the third namespace defines classes that facilitate:
management of data from multiple data sources; and
processing of XML documents.

30.   (Previously Presented) The method as recited in claim 26, wherein the fourth namespace defines classes that facilitate:
programmatic access to configuration settings and handling of errors in configuration files;
application debugging and code execution tracing;
customization of data according to cultural related information;
inputting and outputting of data;
interfacing to network protocols;
viewing loaded types, methods, and fields;
creation, storage and management of various culture-specific resources;
system security and permissions;
installation and running of services;
character encoding;
multi-threaded programming; and
runtime operations.

31.   (Previously Presented) A method implemented at least in part by a computer, comprising:

calling one or more first functions to facilitate browser/server communication;

calling one or more second functions to facilitate construction of client applications;

calling one or more third functions to facilitate connectivity to data sources and XML functionality;

calling one or more fourth functions to access system and runtime resources; and

using a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.

32.   (Previously Presented) The method as recited in claim 31, wherein the first functions comprise functions for construction and use of Web services, temporary caching of resources, initial configuration, creation of controls and pages that will appear as user interfaces, securing Web server applications, and accessing session state values.

33.   (Previously Presented) The method as recited in claim 31, wherein the second functions comprise functions for creation of windowing graphical user interface environments, and graphical functionality.

34.   (Previously Presented) The method as recited in claim 31, wherein the third functions comprise functions for management of data from multiple data sources, and XML processing.

35. (Previously Presented) The method as recited in claim 31, wherein the fourth functions comprise functions for programmatic access to configuration settings, application debugging and code execution tracing, customization of text according to cultural related information, synchronous and asynchronous reading from and writing to data streams and files, creation and management of various culture-specific resources, system security and permissions, installation and running of services, character encoding, and multi-threaded programming.

36. (Previously Presented) A method implemented at least in part by a computer, comprising:

receiving one or more calls to one or more first functions to facilitate browser/server communication;

receiving one or more calls to one or more second functions to facilitate construction of client applications;

receiving one or more calls to one or more third functions to facilitate connectivity to data sources and XML functionality;

receiving one or more calls to one or more fourth functions to access system and runtime resources; and

using a common language runtime layer that can translate Web applications written in different languages into an intermediate language supported by the common runtime layer.

37. (Previously Presented) The method as recited in claim 36, wherein the first functions comprise functions for construction and use of Web services, temporary caching of resources, initial configuration, creation of controls and pages that will appear as user interfaces, securing Web server applications, and accessing session state values.

Application No. 10/087,027

38. (Previously Presented) The method as recited in claim 36, wherein the second functions comprise functions for creation of windowing graphical user interface environments, and graphical functionality.

39. (Previously Presented) The method as recited in claim 36, wherein the third functions comprise functions for management of data from multiple data sources, and XML processing.

40. (Previously Presented) The method as recited in claim 36, wherein the fourth functions comprise functions for programmatic access to configuration settings, application debugging and code execution tracing, customization of text according to cultural related information, synchronous and asynchronous reading from and writing to data streams and files, creation and management of various culture-specific resources, system security and permissions, installation and running of services, character encoding, and multi-threaded programming.

41. (Previously Presented) A method implemented at least in part by a computer, for exposing resources using an application program interface, comprising:

A. exposing a first group of services related to creating Web applications, including:

constructing Web services;

temporary caching resources;

performing initial configuration;

creating controls and Web pages;

enabling security in Web server applications;

accessing session state values;

B. exposing a second group of services related to constructing client applications, including:

creating windowing graphical user interface environments;

enabling graphical functionality;

C. exposing a third group of services related to data and handling XML documents, including:

enabling management of data from multiple data sources;

second functions that enable XML processing.

D. exposing a fourth group of services related to base class libraries, including:

defining various collections of objects;

accessing configuration settings and handling errors in configuration files;

debugging and tracing code execution;

customizing data according to cultural related information;

inputting and outputting of data;

enabling a programming interface to network protocols;

viewing loaded types, methods, and fields;

creating, storing and managing various culture-specific resources;

enabling system security and permissions;

installing and running services;

enabling character encoding;

enabling multi-threaded programming; and

facilitating runtime operations.

(9)   Evidence appendix. None

(10) Related proceedings appendix. None